The Big Book of PowerShell Error Handling

Dave Wyatt

1. The Big Book of PowerShell Error Handling

PowerShell.org
This project can be followed at:
https://www.penflip.com/powershellorg/the-big-book-of-powershell-error-handling
©2015 PowerShell.org

1. The Big Book of PowerShell Error Handling

by Dave Wyatt

Cover design by Nathan Vonnahme

Visit Penflip.com/powershellorg to check for newer editions of this ebook.

This guide is released under the Creative Commons Attribution-NoDerivs 3.0 Unported License. The authors encourage you to redistribute this file as widely as possible, but ask that you do not modify the document.

PowerShell.org ebooks are works-in-progress, and many are curated by members of the community. We encourage you to check back for new editions at least twice a year, by visiting Penflip.com/powershellorg.

You can download this book in a number of different formats (including EPUB, PDF, Microsoft Word and Plain Text) by clicking on the 'Download' link on the right.

PDF Users: Penflip's PDF export often doesn't include the entire ebook content. We've reported this problem to them; in the meantime, please consider using a different format, such as EPUB, when you're downloading the book.

You may register to make corrections, contributions, and other changes to the text - we welcome your contributions! check out our contributor tips and notes before jumping in.

Code for this book, including a spreadsheet of known error types and our analysis, can be downloaded from https://onedrive.live.com/redir?resid=7f868aa697b937fe%21107.

About the Authors

Principal writing in this book was by PowerShell MVP Dave Wyatt.

2. Introduction

Error handling in Windows PowerShell can be a complex topic. The goal of this book ? which is fortunately not as "big" as the name implies ? is to help clarify some of that complexity, and help you do a better and more concise job of handling errors in your scripts.

What is error handling?

When we say that a script "handles" an error, this means it reacts to the error by doing something other than the default behavior. In many programming and scripting languages, the default behavior is simply to output an error message and immediately crash. In PowerShell, it will also output an error message, but will often continue executing code after the error occurred.

Handling errors requires the script's author to anticipate where errors might occur, and to write code to intercept and analyze those errors if and when they happen. This can be a complex and sometimes frustrating topic, particularly in PowerShell. The purpose of this book is to show you the error handling tools PowerShell puts at your disposal, and how best to use them.

How this book is organized

Following this introduction, the book is broken up into four sections. The first two sections are written to assume that you know nothing about PowerShell error handling, and to provide a solid background on the topic. However, there's nothing new in these sections that isn't already covered by the PowerShell help files. If you're already fairly familiar with the ErrorRecord object and the various parameters / variables / statements that are related to reporting and handling errors, you may want to skip straight to sections 3 and 4.

Section 3 is an objective look at how PowerShell's error handling features actually behave, based on the results of some test code I wrote to put it through its paces. The idea was to determine whether there are any functional differences between similar approaches to handling errors ($error versus ErrorVariable, whether to use $_ or not in a catch block, etc.), all of which generated some strong opinions during and after the 2013 Scripting Games.

These tests reveal a couple of tricky bugs, particularly involving the use of ErrorVariable.

Section 4 wraps things up by giving you a more task-oriented view of error handling, taking the findings from section 3 into consideration.

3. Powershell Error Handling Basics

Let's start by getting some of the basics out of the way.

ErrorRecords and Exceptions

In the .NET Framework, on which PowerShell is built, error reporting is largely done by throwing exceptions. Exceptions are .NET Objects which have a base type of System.Exception. These Exception objects contain enough information to communicate all the details of the error to a .NET Framework application (the type of error that occurred, a stack trace of method calls that led to the error, etc.) That alone isn't enough information to provide to a PowerShell script, though; PowerShell has its own stack trace of scripts and function calls which the underlying .NET Framework knows nothing about. It's also important to know which objects had failures, when a single statement or pipeline is capable of producing multiple errors.

For these reasons, PowerShell gives us the ErrorRecord object. ErrorRecords contain a .NET Exception, along with several other pieces of PowerShell-specific information. For example, figure 1.1 shows how you can access the TargetObject, CategoryInfo and InvocationInfo properties of an ErrorRecord object; any one of these might provide information that is useful to your script's error handling logic.